EOforge: Generic Open Framework for Earth Observation Data Processing Systems

Celestino Gomez, Luis M. Gonzalez, José Prieto
GMV – c/ Isaac Newton 11; PTM Tres Cantos, 28760 Madrid
Spain
Tel: +34 918072100; Fax: +34 918072199

cgomez@gmv.es, lgonzalez@gmv.es, jprieto@gmv.es

ABSTRACT

Ground segment payload data segments context is rapidly changing, to follow the increasing EO data needs:

User community requiring EO data is growing from scientific users to a broader community where environmental national and cross boundary agencies, national and local civil security authorities, surveillance departments, public health administration and others are involved. Solving final users needs is the next challenge for EO systems.

Requirements of high performance high reliability system with proven capabilities.

***Multimission**: EO data processing systems are not anymore built to support a single source of data: sensor or mission but to enable merging different mission data sets to apply synergies and answer to the final requirements of performance.*

***Service oriented**: EO data processing is not anymore an isolated process but a complex chain of steps, data fluxes and distributed data transformation is the next coming environment. Different functionalities and capabilities will be remotely offered by the mean of services enabling new functionality to be built on the top of them. Common data formats and interfaces are mandatory to success.*

***Reduction of costs**, is key for any development, a broad spectrum needs to be covered a cost-efficient development is required to progress. More and more complex algorithms are required to be implemented and validated, providing and consolidated environment for execution will reduce the cost.*

***Time to market**. Needs are always prior to solutions. User communities have 'real' and existing problems waiting to be solved. Reduce the time to market for EO data processing system is key to answer users needs.*

***EOforge** is our answer to anticipate this context by bringing the required technology in a generic processing framework, to be the base of any EO data processing development. EOforge provides generic functionality for: data ingestion and data, management, data analysis and visualisation, data processing, data classification and quality assessment and storage and sharing functionality providing open services.*

Gomez, C.; Gonzalez, L.M.; Prieto, J. (2006) EOforge: Generic Open Framework for Earth Observation Data Processing Systems. In *Emerging and Future Technologies for Space Based Operations Support to NATO Military Operations* (pp. P5-1 – P5-10). Meeting Proceedings RTO-MP-RTB-SPSM-001, Poster 5. Neuilly-sur-Seine, France: RTO. Available from: http://www.rto.nato.int/abstracts.asp.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
01 DEC 2006	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
EOforge: Generic Open Framework for Earth Observation Data Processing Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **GMV c/ Isaac Newton 11; PTM Tres Cantos, 28760 Madrid Spain**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM202419., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**10**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

The EOforge Data Processing Framework (DPF) is built based on the following main guidelines:

- Use existing formats for data representation, like HDF, all through the EO Payload Data Segment (PDS).
- Use state of the art technologies: XML, Web Services, Workflow.
- Use open software as much as possible: MySQL, Linux,…
- Be multiplatform, support different platforms: Linux, Windows, and Unix.
- Allow the use of existing interfaces, i.e. MUIS: ESA multimission catalogue for EO products.
- Support last EO systems technologies, i.e. MASS (Multi-Application Support Service System): ESA web services technology standard.
- Based on recognised standards (OGC, ISO, OASIS…)

The framework is designed as to support:

1. The deployment of the framework to any site assuring interoperability between deployed Frameworks
2. On-line plug-in of specific processing to allow the provision of dedicated services to the other users avoiding the flow of huge datasets through the network.
3. Support decentralized management of geodata and geoprocessing
4. Parallel and distributed processing (e.g. multi-process, GRID compliant architecture , …)
5. Extensibility and configurability to allow customisation and the inclusion of new functionality.
6. Multi-instrument and multi-mission processing

EOforge is intended as one of the bricks for the EO data processing standarisation and data sharing and integration initiatives launched within programs like Oxygen and GMES

GMV has successfully proven this technology in different contexts:

- **GlobAEROSOL** (http://www.globaerosol.info), a data processing system generating standard reference multi-year aerosol product over land and water. The end-users service included 80Tbytes of EO data being processed.
- **Gaia Data Access and Analysis Study**, where complex space science algorithms have been integrated in the framework and executed against Tbytes of data.
- **Eumetsat's Polar System Global Ozone Monitoring Experiment–2 (GOME-2)**, where the framework has been used to validate and develop GOME-2 level 1 algorithms.
- **NWC Satellite Applications Facility (SAF)** (Spanish Institute of Meteorology Nowcasting and Very Short Range Forecasting service: http://nwcsaf.inm.es), where the EOforge technology has been used to integrate the different algorithms to obtain the meteorological products delivered within the service and to glue the different components in an operational system.

1. INTRODUCTION

Earth is an integrated system. All the process that influence conditions on the Earth, (ecological, biological, climatological, or geological, are linked and impact one another. In addition all these phenomena clearly influences human activities: communications, transports, security and surveillance, health care and human nominal living. Therefore it is required that Earth Observations systems try to afford it with the same integrated approach: bringing together synergetely the different data sources and stakeholders and cope the main challenge for Earth Observations systems in the next coming years: focus on specific and achievable societal benefits. Answering this challenge requires, sensors, missions, algorithms and data processing and data interchange. EOforge tries to cover the data processing and data interchange, providing an open and extensible framework to glue the EO system integration and to simplify the process to build any processing environment. Consequences of using EOforge are reduce the cost and time to deploy any service and focus on the algorithms development also required by the nex EO challenges.

EOforge open framework is built to cover all the generic functionality of any Earth Observation processing system that can be found in every Payload Data element of any mission Ground Segment, also in higher level of processing stages usually provided by institutions and organisms with the scientific experience and knowledge to produce this data. Covering the common and generic functionality allow, both PDS developers and institutions providing services to abandon technology related development start from a tested environment and focus on the added value elements in the service execution chain.: specific algorithms for the particular case.

EOforge is an open framework to be used from algorithms prototyping (GOME-2 GPP business case further discussed) to final service execution with high requirements of reliability and performance (NWC SAF and GlobAEROSOL service business cases). For algorithms prototyping it is required:

1. Framework covering all the functions for data accessing and data formatting.
2. Plotting capabilities.
3. Intercomparison, including scatters plots.
4. Generic, mathematic, scientific, and logging capabilities.
5. Reporting on algorithms validations.

For a service execution is also needed:

1. Performance: multiprocess capabilities.
2. Data handling and date ingestion.
3. Web enabled services.
4. Integration within other data fluxes and workflows.
5. Service validation.
6. Reliabitity.
7. Integration of legacy application and third part modules.
8. Data flux or functionality flux support.
9. Data publishing.

Algorithms prototyping requirements or instrument level 1 processing requires more low level functionality, however service open execution will require, in addition, open capabilities to enable the complex data fluxes in an integrated EO approach.

2. Design Approach

The following paragraphs describe at a high level functionality of EOforge which tries to enable the different context that could be found in the present and future Earth observation data systems:

- A **Compressive and powerful MMI:** EOforge includes complete functionality at different levels. Simple but complete access to this functionality is very important to deploy and fulfil specific requirements. MMI included functionality is divided in different groups:
    - Configuration. The MMI allows the configuration of any single parameter for the framework and also for the different hosted functionality. Parameters are divided in two groups ones linked to the framework itself (administrative configuration) and functional parameters more concerning with modules hosted within the framework. Parameters are stored by using the MMI in XML files.
    - Logger. The MMI provides access at the different logs generated by the different framework modules.
    - Process monitoring: the MMI allows to launch, stop and check the different processes progress at a given moment.

  - Visualisation: x-y plots, MAPs, time series, spectral plots,… common in EO data system are included as base.
  - Comparison. The MMI provides functionality to plot differences between reference data in the validation phase to check the error distribution.

- **Functionality hosting**. The framework covers common functionality. For specific one the framework provides several ways to integrate additional modules. The integration level can vary and will depend on the particular needs:
  - **Strong integration**: Interfaces are at function/procedure level. The module has access to the whole framework functionality. The module needs to be developed based on interface definition.
  - **Medium integration**. The module providing the hosted functionality is partially decoupled of the system and runs as an independent process. The common functionality accessed is limited to a subset of pre-identified ones called from the module as shared libraries.
  - **Week integration**. In this case there are no shared interfaces at function level but at file/database level. Framework in these approach shall prepare the required input data and handle the output data. Intermediate information is also handled as files and presented in the framework MMI.
  - **Service based integration**. The framework provides functionality to access extern services to complete a defined processing request and also exports different services via open protocols like SOAP to allow external system to access to the framework provided functionality.
- **Performance.** EOforge has been designed to provide higher performance by:
  - By optimizing the code and the design approach.
  - Selecting the more efficient tools and algorithms for the common provided functionality.
  - By efficient data and process handling.
  - By enabling multi-process approaches with different configurations:
    - Pipeline execution.
    - Data division dispatch and data re-assembly.
    - Independent parallel execution.
  - By providing extensibility and resizing capabilities to make use of the available hardware processing software.
- **Webservices.** EOforge can act either as web service client or as a server. Acting as a client the framewrok is able identify processing steps to be defined and a webservice. That means, that complete the workflow could require in some cases make use for external functionality exported by other systems. The system will encapsulate the required data to be send for remote processing, handle the call, gather the result data and send it to the module requesting the call used any of the defined interfaces depending on the integration approach. As a server the framework will export the selected functionality to allow external system to include it as part of their processing workflow.
- **Data fluxes and functionality fluxes support.** When performing remote execution EOforge allows this execution to the handled in different ways (both as a client and as a server):
  - Client sends the input data to be processed and received the processed one. This is the normal way of execution, in this case the server shares the processing functionality and client provides the input data.
  - Client sends the processing functionality. The server provides the input data and processing (hardware) capabilities. This is highly recommended when large amounts of data need to be processed; sending these data set through the network is not practical.
  - Client sends some king of processing identifier and retrieves input data. This a simplified of the nominal case, where the input data is also in the server.

- **Database enabled**: The EOforge framework is powered by a MySQL database for data storage and data retrieval. The databases capabilities allow historical data/parameters to be ingested in a easy way and then retrieved time period for further analysis. The database access export a generic API, enabling any possible parameter to be retrieved. The database provides a data dictionary to enable this feature.
- **Multiplatform (Solaris/Linux/Windows NT):** system has been conceived with multiplatform in mind to cover more usual environments: using multiplatform languages and ANSI standards provides the desired compatibility within the more usual platforms.
- **Extensibility**: In conception the framework is extensible: provides generic functionality and allows additional features to be added. Any new module to the system is part of the core and can be used as other inner module.
- **Robustness**. In any complex system failures, whether caused by software or hardware, are inevitable, therefore the product must be capable of handling them and maintaining a stable state. EOforge has been designed with this in mind and incorporates many such features. They mainly concentrate on data versioning and process integrity. The system must also try to avoid a workflow failing to complete because of some sort of non intrinsic error, like a network failure, a temporary database unavailability, etc.

3. ARCHITECTURE

EOforge is a processing framework, which means a piece of software that is able to manage optimally different kind of (processing) resources. This if the basic and initial set of functionality for which the product has been conceived. The system has evolved, first trying to include more and more functionality that it is often duplicated in the different EO data processing system and in second place to provide means to open this services.

The DPF core concentrates all the required functionality to handle the process management. This core allows the different integration schemas described in the previous section. The DPF core is built using JAVA/JNI techniques. The DPF core also provides capabilities for the different elements modules to communicate depending on the integration level:

- Event based: any module can launch an event to the caught by any of the subscribed modules.
- Blackboard based: Any module can leave any data set in a common placed usually called backboard (same approach as for multi-agents based systems) to be further gather by other modules. This approaches simplifies the data interfaces and assumes that both: module leaving data and module gathering data are able to communicate. A blackboard in the EOforge framework can be either a memory place a directory, a single file,… to be used at different integration levels.
- Pure file based. One module leaves information on one file while the other just waits for this file to be available in the configured place.

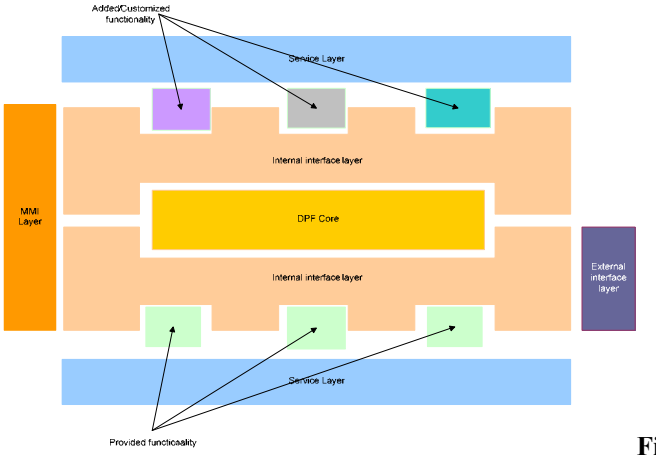

Figure 1 EOforge high level architecture**Multilayer approach.** The DPF exports all the functionally (internally and externally) on a multilayered approach. The number of layers to be used for a particular case will depend on the requested function. For a direct call to an auxiliary mathematical function only the lower layer: function call, will be used. But for a service request through the web will go trough all the data and function interfaces layers in both directions: when processing the request and when sending back the results of the request.

Parameter based interfaces. To simplify the data access to any kind of data the framework provides a generic way to access any kind of data by request based on a parameter ID. Any module can perform any data request without dealing where the data is stored, the framework queries the dictionary, finds the parameter location, identifies way to access the data (by selecting the proper particular data structure format definitions) and performs the required conversions before sending back the requested data. Additional data formats could be added by simple updating the data dictionary placed in the configuration data model.

Three-tierd architecture. The system separates the application from the database and from the presentation layer which allows a higher flexibility and increase of performance. In addition the system decouples part of the application layer in two different layers:

- Web-service layer: where the request are processed and translated to calls to the different framework provided interfaces.
- DPF core layer where the execution is performed and where all the capabilities for data handing and process and modules management are present.

Distributed architecture. In addition to the inner distributed capabilities provided by the WEB services the framework present capabilities to distribute the processing effort by running on different machines.
In order to maximise the use of the available computing resources and balance inequalities in the hardware an on demand strategy is used for distributing the data blocks to the different processing nodes.

MMI. Architecture allows the MMI to be integrated in the same way as any other module, that allows the MMI to use at least in theory the same functionality that any other module. This approach enables the MMI for reporting, database querying, process management,… which is very important in any complex system. Following figures show examples of the MMI capabilities.

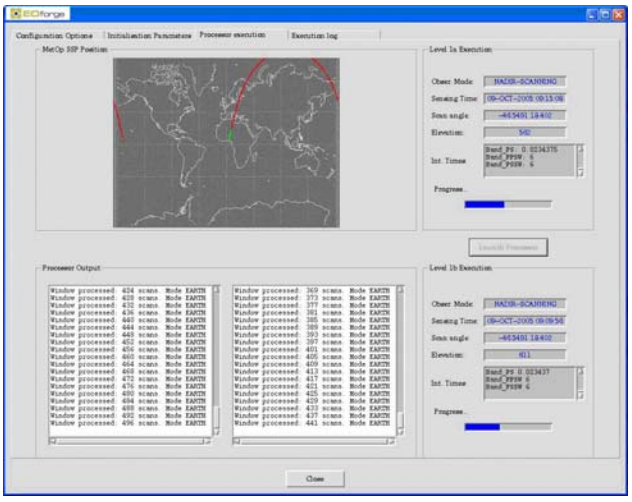

Figure 2 EOforge execution monitoring

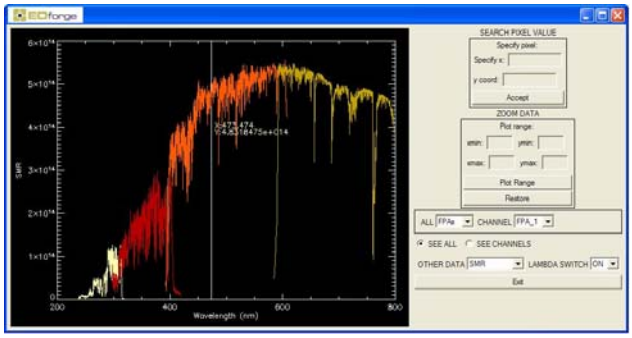

Figure 3 EOforge Plotting capabilities

Support ESA standards for catalogs and services

EOforge architecture provide access to de-facto standards used by the European Space Agency such as:

- MUIS: ESA multimission catalogue for EO products.
- MASS (Multi-Application Support Service System): ESA web services technology standard.
- OGC, ISO, OASIS.

4. TOOLS AND TECHNIQUES.

JAVA

The main technological choice for the product was the selection of the *Java* programming language. Its platform independence and the *RMI* protocol made it ideal for a project which envisioned the same software running in a heterogeneous hardware environment.

Furthermore the Java Native Interface (JNI) guaranteed that algorithms provided by the scientific community in other languages could still be integrated in the system. In this way the system incorporates as shared libraries code written in *C, C++* and *FORTRAN*, which is run over a *C* bridge that transfers the data to and from Java (Figure 4).

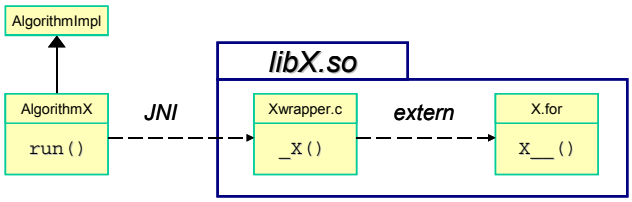

Figure 4: Fortran Algorithm "X" Integration

The threading capability is used for efficient execution of multiple algorithms and stable job control. The data flow between the database and the algorithms is simplified if the latter can be run "in process" avoiding the use of shared memory, pipes or temporary files which would be necessary with external execution.

The API provides great flexibility at runtime which has been taken advantage of by the project in order to accomplish various things:

- Algorithm implementations can define attributes which are used normally as control parameters and these then become configurable from the GUI when preparing an execution plan.
- A layer of abstraction has been developed on top of RMI that allows the remote execution of methods which have not been precompiled for distributed execution. This reduces the development effort and adds flexibility to the system.
- A Query System developed to allow users to examine the database is dynamically extensible and totally database independent. The system analyses in real time the attributes of the persistent objects which exist in the database and permits the creation of queries on combinations of conditions on those attributes. If new objects are added or existing ones modified the query system will work for those without any modifications.
- Create an application that automatically produces *SQL* and *Java* code necessary for the storage of an object in a relational database using *JDBC*. The code generated can deal with navigating single or multiple references, and storing and retrieving arrays, of native and object type, and lists.

UML

The system has been modelled in *UML* and forward engineering has been used for the generation of the basic code skeletons. Scripting in combination with the *UML* model and tool has also enabled the following:

- Generate a database independent interface for a generic object representing data and at the same time add all the methods, and dependencies necessary to a storage implementation object.
- Generate all the code for the accessor methods (*get/set*), implementing lazy loading and navigation of *JDBC* references, and the *equals*, *clone* and *toString* methods. The automated code also copes with arrays, lists and maps.

WEBSERVICES

Web Service Definition Language (WSDL)

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate.

Web service discovery

Web service discovery mechanisms included by the framework includes following characteristics:

Direct description retrieval	Simple aggregate publishing (WS-Inspection)	Advanced directory (UDDI)
Supports some focused and unfocused discovery patterns.	Supports some focused discovery patterns and a significant number of unfocused ones.	Supports a significant number of focused discovery patterns and some unfocused ones.
Dissemination is direct from the source/originator.	Dissemination is direct from the source/originator.	Dissemination is via a third-party.
No overhead.	Moderate overhead.	Moderate overhead for the information owner; high overhead for the directory provider.

WS-Inspection
The WS-Inspection specification provides an XML format for assisting in the inspection of a site for available services and a set of rules for how inspection related information should be made available for consumption.

SOAP
SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP makes extensive use of name spacing and attributes specification tags in almost every element of a message.

6. EOFORGE BUSINESS CASES

Framework technology has been proved in different solutions:

GlobAEROSOL project for ESA, where EOforge enabled technology has been used to build and EO service for final users.

The framework powers a cluster execution and service products access trough the WEB. The GlobAEROSOL service has strong requirements for processing as close to 80Tbytes of data will be processed (11 years of EO data). Service will provide daily, monthly aerosol products to the environmental and meteorological agencies for better model the climatology and for cross-boundary pollution survey and alert. Additional details can be foun in: http://www.globaerosol.info

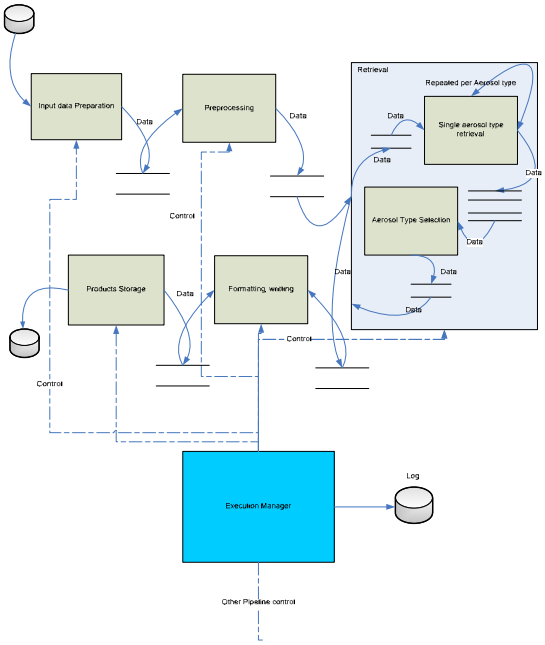

Figure 5 GlobAEROSOL Architecture

GAIA Project

GAIA is an astronomical space observatory which will very accurately measure the positions of a large number of stars. The data set represents ***one of the biggest data reduction problems ever undertaken***. GMV has used the EOforge to develop the GAIA Data Analysis and Access System. Its main objective is to define an efficient, scalable, maintainable and useable system for populating the GAIA mission database from the satellite data stream. This is being done by developing the technical concepts for the system, and then testing them via the development of a prototype database of the system. The solution lies in 2 key design features:

- Algorithms designed to run simultaneously
- Reduction algorithms have access in time & space domains

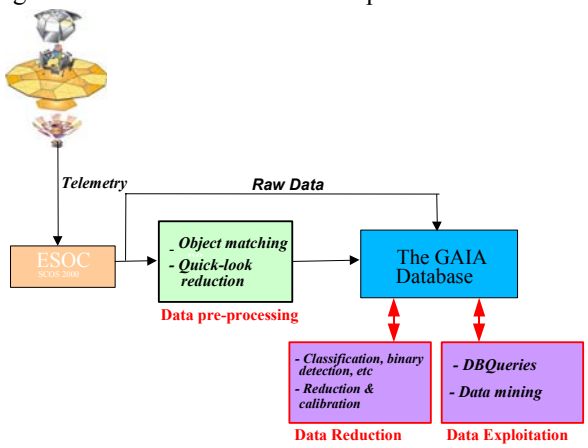

Figure 6 **GAIA Architecture**

SAFNWC

The EUMETSAT Satellite Application Facility for Support to NoWCasting & Very Short Range Forecasting. This Meteorological Facility has been designed in order to obtain the optimum benefit from the new generation of meteorological satellites. GMV has used EOforge technology to power our contribution to the operational service provided by the Spanish Meteorological Institute (INM, http://www.inm.es) and develop a software application to process the MSG and EPS satellite data. This application will generate products on clouds, high-resolution winds, and air mass properties.